```
;; An exercise in reverse Polish notation evaluation suggested
;; by Damir Jamsek.
```

EVENT: Start with the initial **nqthm** theory.

DEFINITION:
length $(l)$
= **if** listp $(l)$ **then** $1 +$ length (cdr $(l)$)
**else** 0 **endif**

DEFINITION: OP1 = '(suc)

DEFINITION: OP2 = '(add mul div mod)

DEFINITION:

op1-form-p $(e)$

$$
\begin{aligned}
= \quad & (\text{listp}\,(e) \\
& \wedge \quad \text{listp}\,(\text{cdr}\,(e)) \\
& \wedge \quad (\text{cddr}\,(e) = \textbf{nil}) \\
& \wedge \quad (\text{car}\,(e) \in \textsc{op1}))
\end{aligned}
$$

Definition:
op2-form-p $(e)$

$$
\begin{aligned}
= \quad & (\text{listp}\,(e) \\
& \wedge \quad \text{listp}\,(\text{cdr}\,(e)) \\
& \wedge \quad \text{listp}\,(\text{cddr}\,(e)) \\
& \wedge \quad (\text{cdddr}\,(e) = \textbf{nil}) \\
& \wedge \quad (\text{car}\,(e) \in \textsc{op2}))
\end{aligned}
$$

Definition:
exp-p $(e)$

$$
\begin{aligned}
= \quad & \textbf{if } e \in \mathbf{N} \textbf{ then t} \\
& \textbf{elseif } \text{op1-form-p}\,(e) \textbf{ then } \text{exp-p}\,(\text{cadr}\,(e)) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \textbf{ then } \text{exp-p}\,(\text{cadr}\,(e)) \wedge \text{exp-p}\,(\text{caddr}\,(e)) \\
& \textbf{else f endif}
\end{aligned}
$$

Definition:
eval-exp $(e)$

$$
\begin{aligned}
= \quad & \textbf{if } e \in \mathbf{N} \textbf{ then } e \\
& \textbf{elseif } \text{op1-form-p}\,(e) \wedge (\text{car}\,(e) = \texttt{'suc}) \\
& \textbf{then } 1 + \text{eval-exp}\,(\text{cadr}\,(e)) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \wedge (\text{car}\,(e) = \texttt{'add}) \\
& \textbf{then } \text{eval-exp}\,(\text{cadr}\,(e)) + \text{eval-exp}\,(\text{caddr}\,(e)) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \wedge (\text{car}\,(e) = \texttt{'mul}) \\
& \textbf{then } \text{eval-exp}\,(\text{cadr}\,(e)) * \text{eval-exp}\,(\text{caddr}\,(e)) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \wedge (\text{car}\,(e) = \texttt{'div}) \\
& \textbf{then } \text{eval-exp}\,(\text{cadr}\,(e)) \div \text{eval-exp}\,(\text{caddr}\,(e)) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \wedge (\text{car}\,(e) = \texttt{'mod}) \\
& \textbf{then } \text{eval-exp}\,(\text{cadr}\,(e)) \textbf{ mod } \text{eval-exp}\,(\text{caddr}\,(e)) \\
& \textbf{else f endif}
\end{aligned}
$$

Definition:
exp-to-rpn $(e)$

$$
\begin{aligned}
= \quad & \textbf{if } e \in \mathbf{N} \textbf{ then } \text{list}\,(e) \\
& \textbf{elseif } \text{op1-form-p}\,(e) \textbf{ then } \text{append}\,(\text{exp-to-rpn}\,(\text{cadr}\,(e)), \text{list}\,(\text{car}\,(e))) \\
& \textbf{elseif } \text{op2-form-p}\,(e) \\
& \textbf{then } \text{append}\,(\text{exp-to-rpn}\,(\text{caddr}\,(e)), \\
& \qquad\qquad \text{append}\,(\text{exp-to-rpn}\,(\text{cadr}\,(e)), \text{list}\,(\text{car}\,(e)))) \\
& \textbf{else nil endif}
\end{aligned}
$$

DEFINITION:
eval-rpn $(r,\ s)$
$=$ **if** listp $(r)$
**then if** car $(r) \in \mathbf{N}$ **then** eval-rpn (cdr $(r)$, cons (car $(r)$, $s$))
**elseif** car $(r)$ = `'suc`
**then** eval-rpn (cdr $(r)$, cons $(1 +$ car $(s)$, cdr $(s)$))
**elseif** car $(r)$ = `'add`
**then** eval-rpn (cdr $(r)$, cons (car $(s)$ + cadr $(s)$, cddr $(s)$))
**elseif** car $(r)$ = `'mul`
**then** eval-rpn (cdr $(r)$, cons (car $(s)$ $*$ cadr $(s)$, cddr $(s)$))
**elseif** car $(r)$ = `'div`
**then** eval-rpn (cdr $(r)$, cons (car $(s)$ $\div$ cadr $(s)$, cddr $(s)$))
**elseif** car $(r)$ = `'mod`
**then** eval-rpn (cdr $(r)$, cons (car $(s)$ **mod** cadr $(s)$, cddr $(s)$))
**else** eval-rpn (cdr $(r)$, $s$) **endif**
**else** $s$ **endif**

THEOREM: eval-rpn-append
eval-rpn (append $(r1,\ r2)$, $s$) = eval-rpn ($r2$, eval-rpn $(r1,\ s)$)

DEFINITION:
l1-ind $(e,\ s)$
$=$ **if** $e \in \mathbf{N}$ **then t**
**elseif** op1-form-p $(e)$ **then** l1-ind (cadr $(e)$, $s$)
**elseif** op2-form-p $(e)$
**then** l1-ind (caddr $(e)$, $s$)
$\wedge$ l1-ind (cadr $(e)$, cons (eval-exp (caddr $(e)$), $s$))
**else t endif**

THEOREM: l1
exp-p $(e)$ $\rightarrow$ (cons (eval-exp $(e)$, $s$) = eval-rpn (exp-to-rpn $(e)$, $s$))

# Index